Eliciting and utilising knowledge for security event log analysis: an association rule mining and automated planning approach

Saad Khan[a,*], Simon Parkinson[a]

[a] *Department of Computer Science, School of Computing and Engineering, University of Huddersfield Queensgate, Huddersfield HD1 3DH, UK*

Abstract

Vulnerability assessment and security configuration activities are heavily reliant on expert knowledge. This requirement often results in many systems being left insecure due to a lack of analysis expertise and access to specialist resources. It has long been known that a system's event logs provide historical information depicting potential security breaches, as well as recording configuration activities. However, identifying and utilising knowledge within the event logs is challenging for the non-expert. In this paper, a novel technique is developed to process security event logs of a computer that has been assessed and configured by a security professional, extract key domain knowledge indicative of their expert decision making, and automatically apply learnt knowledge to previously unseen systems by non-experts.

The technique converts event log entries into an object-based model and dynamically extracts associative rules. The rules are further improved in terms of quality using a temporal metric to autonomously establish temporal-association rules and acquire a domain model of expert configuration tasks. The acquired domain model and problem instance generated from a previously unseen system can then be used to produce a plan-of-action, which can be exploited by non-professionals to improve their system's security. Empirical analysis is subsequently performed on 20 event logs, where identified plan traces are discussed in terms of accuracy and performance.

Keywords: Security Event Logs, Association Rule Mining, Temporal ordering, Automated Knowledge Acquisition, Automated Planning

1. Introduction

Most organisations are facing security threats exposed by their digital infrastructure, and given the increasing size and critical nature of their business operations, there is a need to pro-actively identify and mitigate security vulnerabilities. This process requires expert knowledge of the latest security threats and how they can be mitigated. Such knowledge is in short supply, expensive and sometimes inaccessible, but is essential to keeping the system secure against malicious attacks.

The lack of expertise can potentially be resolved using computational intelligence to autonomously generate security evaluation and mitigation plans. In this paper, an automated solution is presented that can extract security actions performed on a system using only event log data. An event log entry contains information regarding the activities within the system, generated by either a user, application or the system itself (Schauland and Jacobs, 2016). Event logs are beneficial for tracking (step-by-step) state changes in the system (e.g. creating or deleting users), as well as troubleshooting for problems. This solution builds up the knowledge by discovering the chains of event relationships to represent a specific human expert's action. The relationships are defined as temporal-associations (Ale and Rossi, 2000), which are achieved

*Corresponding author
Email addresses: saad.khan@hud.ac.uk (Saad Khan), s.parkinson@hud.ac.uk (Simon Parkinson)

by incorporating notion of time to the frequent co-occurrences in given dataset. The solution is currently aimed at the Microsoft Windows operating systems (OS) (Simache et al., 2002), but is operating system independent and can easily be adapted for other systems. The data residing within Microsoft event log entries is structured and contains object-value pairs. A default set of classes exist in the .NET framework for parsing and processing, and this facilitates an easier development and creation of testing environment for the proposed approach. Future extensions of this work will add more features focussing on other, widely used Linux-based systems, such as OpenStack cloud (Mushtaq et al., 2017) and Fog computing platforms (Khan et al., 2017).

The security event logs provide detailed state information on a systems security controls. For example, login and logout attempts, permissions change, activity based on firewall rules, cryptographic operations and many other security-related events. The system activity is recorded based on the predefined security and configuration policies by the administrator. According to Microsoft documentation[1], each entry contains a series of objects that define a particular occurring of an event. For example: a unique identifier (ID), event source, user account, time-stamp, machine name, and application- and service-specific objects.

Security event log provide knowledge that describes both unauthorised activities and configuration events. If correct correlations are discovered, the relationship among the events can be used to model the identification of security problems and implementation of remedial actions. It is also likely that a security configuration activity will generate a sequence of security and configuration events according to a specified audit policy, forming a complex chain of events. There is a great potential to learn expert knowledge from event logs, and to the best of the author's knowledge, there has been little research into automatically capturing and exploiting expert knowledge from this source. Besides event logs, there are various other data sources available, such as research articles, community forums, lecture audio etc. that can provide useful and practical knowledge. Few research studies are available that can extract certain information by processing unstructured data (text) using Markov Logic Networks (Natarajan et al., 2017) and intelligent pattern recognition (Mikhaylov et al., 2017), but it would require further advancements in the domain of natural language processing to extract applicable knowledge. Event log entries, as shown in later sections, contain structured pairwise objects that can be easily processed to generate accurate results.

It is of great value to automatically identify such chains of events. Consider a scenario where a system administrator is maintaining a server. A user illegally attempts to access a file named *abc.txt* and successfully viewed it. To document this activity, an event will be recorded with the event type ID as 4663 depicting *An attempt was made to access an object*. Assuming the administrator would notice that the user is not supposed to access any file on this server and their permissions should be immediately removed. The administrator would then modify the network share permissions, and as a result an event log entry will be created with the event type ID of 5143, detailing *A network share object was modified*. The event logging mechanism generates this event in a way, where it includes the record of old permissions along with the new one. Many additional event log entries will be generated in between 4663 and 5143, and therefore it would require expert knowledge, time and effort to discover this link. Despite being a simple example, it signifies the fact that managing security requires experience and expert knowledge. A system that can automatically extract a (temporal) association between 4663 and 5143 would be beneficial for the non-experts to directly look for such event entries and determine if any user is exercising more than the required set of permissions. An open-source and cross-platform tool, called Simple Event Correlator (SEC), exists for this purpose that performs rule-based event correlation (Vaarandi, 2002). The tool has various significant features and is available for download[2] as well. However, it lacks in automation and is based on pre-defined knowledge base of hard-coded rules used to detect the event relations. Unlike SEC, the proposed solution is based on fully unsupervised and automated approach, which is its key contribution.

The research hypothesis addressed in this paper is whether it is possible to identify security-related actions performed by an expert and allow non-experts to utilise the knowledge to audit and improve the security of their systems. In exploring this hypothesis, an unsupervised knowledge acquisition technique for identifying

[1]Microsoft Windows documentation on Event Logging and Viewing: https://msdn.microsoft.com/en-us/library/bb726966.aspx

[2]https://sourceforge.net/projects/simple-evcorr/

temporal-associations among event log entries is developed. It utilises correlation mining processes due to their known suitability in finding such relationships (Angelov, 2013). The identified correlation rules are then used to produce chains of events that are ordered based on a temporal metric and form a domain action model, which is subsequently used by an Automated Planning (AP) algorithm to determine a course of action on a previously unseen system. The primary contributions are:

- **A novel technique to extract and represent knowledge from event log entries**. This technique models the event logs into *object-based* entries, which are utilised by a association mining process for identifying correlation rules. Those rules are then converted and validated into chains of temporal-association relationships (based on the given event log entries), and then encoded into domain model.
- **A novel software tool for Microsoft's event log mining**. The developed software tool enables expert users to automatically extract domain models from their machines and store them for other non-expert users. The automation of this process greatly reduces the knowledge gathering time.
- **A software tool for parsing planner output.** The action plan output generated by the planner using domain and problem files is difficult for non-experts to comprehend. This tool parses and represents the planner output in a human-friendly format for better understanding.
- **Empirical analysis**. The significance of the proposed technique and tool is then evaluated on 20 event logs datasets from live system, multi-user network of machines to determine overall performance and accuracy.

The paper is organised as follows: The first section surveys existing domain learning techniques, AP and correlation mining techniques. The next section presents how to determine the temporal-associations among event log entries. The following section shows the process of creating a domain action model from temporal-association rules. Finally, the last section includes an empirical analysis of solution using live system event logs.

2. Related Work

Automated Planning (AP) is a process of selecting and organising purposeful actions in achieving expected outcomes (Ghallab et al., 2004). Recent studies show that AP is being used in numerous cyber security related areas, mainly for offensive attack planning (Amos-Binks et al., 2017). Different approaches have been developed in this area, for example, model-based simulation of penetration testing (Hoffmann, 2015), generating adversarial courses of actions (Riabov et al., 2016) and security threat mitigation plans (Backes et al., 2017). The Planning Domain Definition Language (PDDL) and its extensions are commonly used for encoding domain knowledge (McDermott et al., 1998). Given the knowledge is encoded into PDDL domain action model and a problem file, various planners are available based on general problem-solving techniques. They can generate plans with respect to quality and time-based constraints (Valenzano et al., 2010). The process of generating domain models requires additional expertise in modelling complex domain actions. This has resulted in a pursuit of autonomously processing available data sources to learn domain knowledge.

Within the AP community, significant research has been performed to acquire a domain action model from various knowledge sources, such as plan traces and human experts (Shah et al., 2013). Creating domain models manually from these sources is a time-consuming and error-prone task. This is further exacerbated by the applications of domain-independent planners resulting in the need to autonomously acquire and learn domain knowledge. Autonomous knowledge acquisition (KA) is an efficient process and has facilitated the exploitation of AP in new areas. The computer generated plans have shown more efficiency as they explore wide array of possibilities (Khan and Parkinson, 2017). However, there are significant challenges, such as gathering correct and complete knowledge, adapting the incremental growth of knowledge and encoding the knowledge into efficient and error-free action schemas.

When developing a domain model, knowledge can be collected in both manual and autonomous manners. Various manual tools are available to assist in constructing domain models. For example, itSIMPLE (Vaquero

et al., 2012), Simultaneous Learning and Filtering or SLAF (Shahaf and Amir, 2006), Action-Relation Modelling System or ARMS (Wu et al., 2005) and LOCM with Optimised Plans or LOP (Gregory and Cresswell, 2015). One significant challenge faced in autonomous knowledge acquisition is that of learning from incomplete and noisy data sources. Researchers have produced tools such as Opmaker (McCluskey et al., 2009), Learning object-centric models or LOCM (Cresswell et al., 2013) and Action-Model Acquisition from Noisy plan traces or AMAN (Zhuo and Kambhampati, 2013). The learning process of these tools is dynamic and sometimes take additional information (such as partial domain model and uncertainty factors) to output a full domain model. Despite the advantages, limitations do exist over the compatibility with domains having static facts, which remain true throughout the planning process and does not change (Botea et al., 2005). Recent advancements are starting to address this issue, such as in Automated Static Constraint Learner or ASCol (Jilani et al., 2015). This exploits a directed graph representation of operators' arguments in a plan trace and discovers pairwise relations among predicates.

The input (plan traces) used in these systems are gathered from goal-based solutions and manual observations or experiments. The major drawback of state-of-the-art solutions is the need for pre-existing domain models and human assistance. They lack in assigning meaningful names to all learnt actions, predicates and objects types. All existing domain learning solutions process plan traces, and as such performing autonomous learning in a previously unseen application area would first require the construction of a plan trace. In most domain engineering processes, this is not feasible because creating a well-formed plan trace could be viewed as almost as challenging as constructing a domain model. In addition, it would also require expert knowledge, something which can not be guaranteed to be available. For example, techniques like crowd-sourcing models (Zhuo, 2015) and involving human experts (Long et al., 2009) also require large amount of effort and human-error/conflict resolution. This motivates the research hypothesis of developing an autonomous approach to learn and utilise domain knowledge for implementing secure protocols directly from data streams, without prior knowledge and human aid. The extracted plan trace can help non-experts and experts alike for the betterment of security in any new or previously unseen machines.

Considering the importance of ascertaining a domain model, this research pursues the potential data mining techniques to discover relationships among a diverse set of event log entries. It explores the use of association rule mining (ARM) techniques to identify relation among security events (Hipp et al., 2000). It utilises pattern analysis technique for revealing strong co-occurrence rules among seemingly unrelated items of event log entries. The first ARM algorithm (Agrawal et al., 1993) was developed to identify regular co-occurrences between the products in large-scale transaction data. The data was recorded by point-of-sale (POS) systems in supermarkets. The problem is that it can only have one consequence statement in the rule i.e., it can generate $X \cap Y \rightarrow Z$, not $X \rightarrow Y \cap Z$ rules. Following this, two new breath-first search based algorithms were introduced called *Apriori* and *AprioriTid* (Agrawal et al., 1994). Their main target was to improve the performance of association mining process in terms of execution time. Apriori was the first algorithm to pioneer support value based pruning and avoid the exponential growth of possible itemsets. Their general idea was, given an itemset $ABCD$, first extract their subsets ABC, AB etc. If a subset does not generate a rule, further subsets of ABC can be avoided based on the Monotonicity Principle (Gal, 2003). Similarly, if any itemset is frequent, then every subset of the itemset will also be frequent (also know as the Apriori Principle). The itemsets are stored in a hash tree format to further increase the performance (Park et al., 1995). ARM has been widely exploited in many different application areas, and Apriori is a commonly used algorithm (Agrawal et al., 1994). It uses efficient pruning scheme to prevent exponential growth of candidate frequent itemsets. In other research, the authors use a modified variant of ARM to identify irregular file system permissions (Parkinson et al., 2016). This solution discovers the least frequent rules as oppose to finding strongly correlated rules like in most applications of ARM. This allows them to identify irregular or rather suspicious permissions that should be eliminated to improve data security.

It should be noted here that the presence of association rules among various mutually exclusive items of the dataset do not necessarily imply they are connected in real time (Morgan and Winship, 2014). The rules are subjective in nature and only based on the given data (i.e., the results are local, not universal). A rule only depicts that there is a strong co-occurrence (correlation) between antecedent and consequent items. The quality of association rules can be improved as evident by many studies, providing a meaningful mechanism to identify correlation. The temporality or time-based ordering can be used to identify

or formulate interesting association rules among items (Box et al., 2015). It provides an additional piece of evidence for the relationship to exist. For example, determining that event-A always happens after the event-B can help establish a higher confidence (Bechlivanidis and Lagnado, 2016). Different research studies have proposed tree-based algorithms that can perform temporal association rule mining and discover local patterns (Verma et al., 2005; Wang et al., 2018). The algorithms effectively reduce the computational cost by skipping the candidate item set generation phase. Another paper proposed Multi-level Intensive Subset Learning (MIST) algorithm, which applies temporality to correlation rules from financial time series (Tan et al., 2015). The algorithm successfully identified profitable trades in terms of association rules by uncovering hidden knowledge. Another study presented a new algorithm, called T-Apriori (Liang et al., 2005). The algorithm includes time constraint and is used for analysing ordered ecological events set. However, according to the paper, the efficiency of the algorithm has not been tested on larger datasets. It also requires an additional layer of data-preprocessing using K-Means clustering algorithm, which might reduce the performance of overall solution.

Another paper presents an algorithm, called ARMADA that can discover time-dependent correlations or patterns within large datasets (Winarko and Roddick, 2007). It is claimed that the generated temporal association rules provides richer rules in terms of semantics. However, this algorithm was only tested on synthetic datasets. A different study determined that the incorporation of temporal dependencies in rule mining process can group and (re)arrange rule items into correct semantic sequences (Roddick and Spiliopoulou, 2002). Another study presents a conversion methodology from association to temporal-association rules given the spatio-temporal data (Bleisch et al., 2014). It requires input of states and events of interest. The approach has been successfully tested on live systems data regarding the fish movement in a river. There are also other mining techniques where temporal dependencies are considered of key importance, for example, the Bradford Hill (Miklossy, 2011) and Granger (Sliva et al., 2015) models.

Another research work created a static approach to determine the reason behind certain system failures (Sahoo et al., 2003). It performed correlation mining on a limited set of event log entries and used time-series methods and rule-based classification to identify patterns. Another paper (Nazerfard et al., 2011) presents a solution for assisted living using temporal-association rule mining. The algorithm is used to recognise and build predictive activity models. Despite of having significant advantages, the aforementioned approaches do not provide a suitable solution for identifying temporal-associations in a completely unknown raw data streams. They either rely on manual input or operates in a constrained and predictable environment to define connections. Some of the solutions require pre-configuration and pre-processing layers, which are not suited for any large, complex models. Also, the requirement of manual input again leads towards needing the human experts, which as discussed before are in short supply. Moreover the existing techniques are not capable of determining the origin (first event) of sequence of multiple correlated events and do not provide the basis to develop an action model.

3. Mining Event Relationships

This section details the processing of event logs to identify temporal-association rules. A brief overview of the five-stage process is:

1. Convert event log entries to an **object-based model**;
2. Identify **object-based rules** among the entries of model;
3. Determine **event-based rules** where any two events are associated by the object-based rules;
4. Improve the quality of relationships based on events' temporal order and filtering insignificant relationships; and
5. Expand on the individual relationships in **forming and validating sequences of events**.

3.1. Preparing Object-based Model

In the following discussion D is used to model the set of event entries, where $D = \{E_1, E_2, ..., E_N\}$. The event, $E = \{ID,O\}$, where ID is a numeric event type ID, and O is the set of unique objects, such that $O = \{O_1, O_2, ..., O_N\}$. Each entry, E_i, contains its relevant objects from I to represent an occurrence of

event. Note that the event type IDs are not part of D as it is only used for mining object-based correlation rules. For example, $D = \{E_1, E_2, E_3, E_4\}$, where:

$$E_1 = \{\{4567\},\{User1, Win7, Port : 53176, NTLM\}\}$$
$$E_2 = \{\{1234\},\{User1, ReadEA, svchost.exe, IKE\}\}$$
$$E_3 = \{\{2345\},\{User2, ReadEA, System, NTLM\}\}$$
$$E_4 = \{\{5678\},\{User2, Win7, NtLmSsp, Winlogon\}\}$$

In this trivial example, each entry has 4 objects along with their event type IDs. The objects of an entry are separated by a comma and represent the information such as user name, OS, applications and network ports. Each entry starts with the corresponding event type ID. It should be noted that the original event log entries are much more diverse and contain more objects. This example is a simplified representation to show the working of proposed solution. Events occur over the duration that a system is running and they appear in sequence, although concurrency of different processes can generate intricate entries. A sequential example is presented in Figure 1, where the events occur over a 20 second period. Each entry contains a timestamp to denote when this log entry was created. The example provided in Figure 1 is a synthetic example used throughout the paper to help communicate the technical approach.

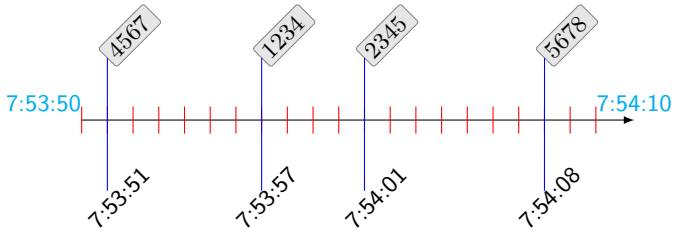

Figure 1: The visual time-line representation of D covering a time span of 20 seconds.

3.2. Object-based Association Rules

This section presents a technique to determine correlation rules among event log entries using ARM processes. The purpose of this section is to identify such objects that inform of events, which are likely to occur together. ARM is utilised as a method for describing, analysing and presenting association rules (ARs) that satisfy the condition of (strong) co-occurrence. Here the ARs are discovered in tabular dataset of objects that uses different measures to determine interesting data (Piatetsky-Shapiro, 1991). ARM consists of two fundamental steps, which are taken to extract usable ARs (Karthikeyan and Ravikumar, 2014):

- *Frequent itemset* – find the set of items that appears more than a certain number of times (defined by the user) in data; and
- *Correlation* – identify strong co-occurrences among the frequent itemset, where the degree of strength is flexible and defined by the user.

Considering D is the set of event log entries, the following three steps would be taken in order to discover interesting relationships (Ishibuchi et al., 2007). In the following explanation, a set I is introduced which represents a total set of unique objects, $I = \{O_1,O_2,...,O_n\}$ acquired from event entries. Both X and Y represent individual objects extracted from I. More specifically, $X = O_i$ and $Y = O_{i+1}$.

Step 1 (Association rule). *$X \rightarrow Y$ is an Association Rule (AR), where X and Y contain one or more objects each from I. X is the LHS (left-hand side) or body and Y is RHS (right-hand side) or head of rule. Both X and Y are disjoint objects, i.e., $X \cap Y = \emptyset$. $X \rightarrow Y$ means whenever E contains X then E probably contains Y too. In the continuing example, an AR would be $\{User1\} \rightarrow \{Win7\}$.*

Step 2 (Support of an AR). *The support (supp(X)) of an AR is defined as the percentage of event log entries that contain both X and Y. It can also be defined as the probability $P(X \cup Y)$. The example AR has a support of $1/4 = 0.25$, since it occurs in 1 out of 4 event log entries.*

Step 3 (Confidence of an AR). *The confidence of an AR is defined as the ratio between the number of transactions that contain $X \cup Y$ and the number of transactions that contain X. It can also be defined as $P(X \cup Y|X) = P(X \cup Y)/P(X) = conf(X \rightarrow Y)$. In the example, the confidence (supp($X \subset Y$)/supp(X)) of rule $\{User1\} \rightarrow \{Win7\}$ would be equal to $0.25/0.5 = 0.5$. This shows that for 50% of the event log entries that contains $Win7$ are related to $User1$.*

Researchers have demonstrated that ARM generates a large quantity of rules and has a complexity of $O(N^2)$ where N is the total number of unique elements across all log entries (Sarma and Mahanta, 2012). Every ARM algorithm takes minimum support ($minsup$) and confidence ($minconf$) threshold values. Their purpose is to find all those rules having support $\geqslant minsup$ and confidence $\geqslant minconf$. Although any rule with a greater support count and confidence value is deemed relevant (Singer and Willett, 2003), they still might not be interesting to the users.

3.3. Event-based Association Rules

At this stage, correlations among event's objects have been identified. However it is then necessary to translate these relationships to identify connections among actual events based on their type IDs. The object-based rules are in the form of $X \rightarrow Y$, where X and Y contains at least one object each. Converting them into event-based rules requires searching the objects from both X and Y individually within all event log entries. This process starts by locating one or more matches between the objects of X and Y and a particular segment of event log entries. Only those events are selected for matching, whose support $\geqslant minsup$. Upon finding the match, the objects are replaced by respective 'event type IDs' to form a new set of event-based rules. The process also includes a condition that prevents the occurrence of same event type ID on both LHS and RHS of rule. In the continuing example, consider the object-based rule $\{User1\} \rightarrow \{Win7\}$. The $User1$ is found in the log entries with event type ID as 1234 and 4567, whilst $Win7$ is found in 4567 and 5678. As 4567 is already found in LHS, the resultant event-based rule would be $\{1234, 4567\}$ — $\{5678\}$.

At this stage, the correlation rules among events are grouped together, but they lack sequential ordering. It is clear that the relationship do exist between the events, as evident from the object-based association rules, however their appropriate arrangement is yet to discover. From the given information, there is no reasonable way of identifying which event of a rule occurred first, second and so on in terms of time. As we are aiming to provision step-by-step instructions to the users, the events' ordering is a crucial part of process. Also, due to ambiguous and unknown sequence, a bidirectional symbol (—) is used to represent the undefined direction in the event-based association rules.

3.4. Temporal-Association Relationships

Now that the event-based correlation rules have been established, it is then necessary to determine the ordering within each association relationship. The proposed algorithm utilises a temporal metric based on the timestamp of event log entries to determine when certain event was generated, which is then used to determine the order of event type IDs in all rules. The metric is referred as *temporality* from now onwards. The event logging mechanism always attach timestamp with the event log entry. Similar approaches has been successfully applied in various domains such as cognitive, affective and social Neuroscience (Treur, 2016), languages translation paradigm for active sentences (Bettinsoli et al., 2015) and others. The event logs are always generated in linear order, and in case of windows logging mechanism, it is not allowed to modify or delete specific event entries. Considering this and existing literature, it is shown that the temporality can play an integral role in extracting quality relationships in such scenarios.

The technique is presented in Algorithm 1. It starts by iterating over event-based rules on line 3 and determines all pairwise subset combinations between the items of LHS and RHS on line 5. For a rule (x_1, x_2) — (y_1, y_2), the subset would be $(x_1$ — $y_1)$, $(x_1$ — $y_2)$, $(x_2$ — $y_1)$ and $(x_2$ — $y_2)$. The total number of subsets

Algorithm 1 Identifying and filtering temporal-association rules.

Input: Set of event-based rules $R = \{r_1, ..., r_n\}$, where $r = (r_x \text{ — } r_y)$, and r_x and r_y consist of at-least one $EventID$ each

Input: Set of ordered event log entries D containing 2-tuple of $EventIDs$ and their corresponding objects, which were used in creating object-based associative rules

Output: Set of temporal-association rules $C = \{(c_1, TAA_1), ..., (c_n, TAA_n)\}$, where $c = (c_x \rightarrow c_y)$ and c_x results in c_y event and TAA is the temporal accuracy of relationship

procedure TEMPORAL-ASSOCIATION-RELATIONSHIP
  Initialise $C \leftarrow \emptyset$
  for all $r_i \in R$ **do**
    $(r_x, r_y) \leftarrow r_i$
    for all $EventID_x, EventID_y \in r_x, r_y$ **do**
      $PosE_x \leftarrow$ **GetIndicies**$(EventID_x, D)$
      $PosE_y \leftarrow$ **GetIndicies**$(EventID_y, D)$
      $t_f \leftarrow$ **Count**$(\forall x \in PosE_x < (\forall y \in PosE_y))$
      $t_s \leftarrow$ **Count**$(\forall y \in PosE_y < (\forall x \in PosE_x))$
      Initialise $TAA \leftarrow 0$
      Initialise $direction \leftarrow 0$
      if $t_f > t_s$ **then**
        $TAA \leftarrow t_f/(t_f + t_s) \times 100$
        $direction \leftarrow 1$ ▷ means $X \rightarrow Y$
      else if $t_f < t_s$ **then**
        $TAA \leftarrow t_s/(t_f + t_s) \times 100$
        $direction \leftarrow -1$ ▷ means $Y \rightarrow X$
      end if
      if $TAA \geq 50$ **and** $direction$ is 1 **then**
        $C.Add((EventID_x, EventID_y), TAA)$
      end if
      if $TAA \geq 50$ **and** $direction$ is -1 **then**
        $C.Add((EventID_y, EventID_x), TAA)$
      end if
    end for
  end for
end procedure

is the product of the number of elements on LHS and RHS, which is $2 \times 2 = 4$ in the example. The main reason for finding the subsets is to determine if there exists a temporal link between each pair of correlated events. It should be noted here that processing each subset combination of all rules is computationally expensive. However, it is a trade-off between quality of rules and the time and effort required to generate them. Processing each subset combination will reduce the risk of missing any interesting rule. Upon finding the appropriate ordering, the subset combinations of the items-set in the rule would be considered for further processing, or else it would be discarded. From the ongoing example, the rule (1234,4567 — 5678) will be divided into two subsets (1234 — 5678) and (4567 — 5678).

The next step is to determine the *temporal-association accuracy* (TAA) of all subsets from each event-based rule. It will facilitate the conversion of correlation rule into temporal-association connection. The TAA value ranges from $0 - 100$. It depicts the number of times a certain relationship was found accurate, i.e., correct event ordering based on the temporal sequencing of events. It is also demonstrated in (Pearl, 2009) that the temporal ordering of entities can provide beneficial results in inferring the practical and useful event connections. Processing events based on their temporal sequence does induce a degree of uncertainly as we do not know how reliable the temporal sequence is. For example, it could be possible that the event

logging processes operate at a low system priority and the process of raising an event may be queued during heavy processing.

For every subset, the indices of event type ID from LHS (line 6), as well as RHS (line 7) are determined and saved in $PosE_x$ and $PosE_y$ lists, respectively. The indices are acquired from database D and sorted in timely order. After that, by comparing the elements of $PosE_x$ and $PosE_y$ lists, calculate the number of times each LHS event occurred before every RHS event and save the count in t_f, as shown in line 8. From a development perspective, this comparison is performed using nested iteration. Similarly, determine how many times each RHS event occurred before every LHS event and count the value in t_s (line 9). If the value of t_f is found to be greater than the value of t_s on line 12, it means the LHS event (mostly) occurred before the RHS and the direction of rule will be $X \rightarrow Y$. Otherwise the direction will be $Y \rightarrow X$ due to else condition on line 15. In case t_f and t_s are equal, the correlation rule becomes ambiguous, and the subset rule is ignored. The reason behind this two-way comparison is to find the direction based on temporal validity and individually establish the TAA of every subset. The TAA value is calculated as a percentage of times any correlation rule was found correct as shown in lines 13 and 16. This results in all subsets having TAA value of 50% or above and are accumulated in a set C for further processing along with the newly found directions (lines 20 and 23).

The 50% TAA threshold value means that only those temporal-association rules will be selected, where half of the times or more, LHS event occurred before RHS or vice versa. The only purpose of TAA value is to show the accuracy of respective relationship, regardless of what it represents. From a deterministic point of view, any rule with TAA less than 100% might be spurious, but 50% threshold value was applied for two major reasons. First, higher threshold values lead to empty results in some datasets, which maybe due to the large amount of noise consisting of routine, repetitive log entries. This produces a relatively inconsistent object-model and therefore makes it difficult to obtain rules with 100% TAA. Secondly, rules with a lower TAA value can still provide beneficial knowledge to user. Hence we selected this threshold value to tolerate somewhat 'inaccurate' rules, rather than having none at all. The threshold value can be changed; however, 50% is sufficient based on our empirical analysis.

As an example, assume there are 200 event log entries instead of 4 in the previous example. Considering the rule subset (1234 — 5678), the event type ID from LHS (1234) might potentially occur at $E_9, E_{36}, E_{59}, E_{73}, E_{105}$ and the event type ID from RHS (5678) at $E_{21}, E_{43}, E_{57}, E_{88}, E_{112}$. The t_f and t_s values of (1234 — 5678) would be 14 and 11 respectively. As the t_f value is greater than t_s, the final temporal-association would be $(1234 \rightarrow 5678)$ with $14/(14+11) \times 100 = 56\%$ TAA.

3.5. Forming and Validating Sequences of Events

This section describes the process of formulating temporal-association relationships into one or more chains or sequences of related events. The sequences are generated similar to a lattice-based approach that outputs high utility generic basic (HGB) to describe a complete set of events triggered to conduct an expert's security-related activity (Mai et al., 2017). It starts by iterating over all subsets $\{(c_{x_0} \rightarrow c_{y_0}), \ldots, (c_{x_n} \rightarrow c_{y_n})\} \in C$ and combine LHS event type IDs of those subsets whose RHS event type IDs are the same. This outputs a group, G, containing the combined subsets. Each member $g \in G$ will have one or more event type IDs on LHS linking a single event type ID on RHS. The purpose is to connect all those events together, which lead towards the same goal. Although at this point, the combined events represent the unordered steps to perform one or more particular actions. From the previous example, assume the TAA value of other subset $4567 \rightarrow 5678$ is 70%. In that case, $g = (1234, 4567 \rightarrow 5678)$, and G would be equal to g as there are only two subsets in total. Set G implies that all events from LHS $(1234, 4567)$ lead towards a single event on RHS (5678).

The next step is to create an ordered set of events within every $g \in G$, so that the sequence of correlated events can be formulated into a chain. First step is to find those two event entries having a maximum time difference. It is assumed that those two events will mark the starting and ending events of action(s) that were performed on the underlying system. Similarly, identify the second to last event based on the time that it happened before the ending event. Repeat the process until the last event is reached. This process will arrange the event entries within g in terms of time, therefore defining the initial set of temporal associations. Referring back to Figure 1, it can be seen that the time difference between 1234 and 5678 is

11 seconds and 4567 and 5678 is 17 seconds. This means 4567 occurred 6 seconds before 1234, which is why it would be considered a starting point. The final chain or sequence of temporally associated events would be (4567 → 1234 → 5678).

The next step is to validate the extracted sequences of events. This process gathers and determines a TAA value for each pair of the sequence. After that, it takes an average of all TAA values, which is considered an overall TAA for the sequence. Continuing the example, TAA values will be calculated between 4567 → 1234 and 1234 → 5678. Assume the values are 70% and 80% respectively, the overall TAA of the sequence will be $(70 + 80)/2 = 75\%$. In case the TAA value of a sequence is lower than 50%, the rule will be discarded. Hence the chains of events are formulated using frequent pattern observation of events (ARM) and temporal evidence yielded by Algorithm 1.

4. Application of Automated Planning

At this stage, the sequences of temporally associated events that are represented by event type ID have been established. It would require manual effort to understand the sequences in terms of administrative security actions and utilise the knowledge in practical environment. Hence the final stage of the research is to model these relationships into appropriate domain actions and utilise AP techniques. The transformation of these relationships into the domain action model performs the representation of knowledge and increases the usability as many AP algorithms and their practical implementations are available.

AP algorithms implement a state-transition system is a 3-tuple $\sum = (S, A, \rightarrow)$ where $S = (s_1, s_2, \dots)$ is a finite set of states, $A = (a_1, a_2, \dots)$ is a finite set of actions, and $\rightarrow$: $S \times A \rightarrow 2^s$ is a state-transition function. A solution P is a sequence of actions $(a_1, a_2, \dots, a_k)$ corresponding to a sequence of state transitions $(s_1, s_2, \dots, s_k)$ such that $s_1 =\!\!\rightarrow (s_0, a_1)$, $\dots, s_k =\!\!\rightarrow (s_{k-1}, a_k)$, and s_k is the goal state. A system's configuration is represented by a set of first-order predicates which are subsequently modified through the execution of each action, $a = \{pre+, pre-, eff+, eff-\}$, where $pre+$ and $pre-$ are the first-order preconditions that are positive and negative in the action's precondition list, respectively. Similarly, $eff+$ and $eff-$ are the action's positive and negative effects.

4.1. Modelling

This section presents the process of encoding of temporal-association rules into an action-based domain model using the PDDL, without any human intervention. PDDL is a standardised format supported by numerous AP applications. A PDDL domain model consists of types of objects, predicates modelling facts, functions to handle numeric functions and actions. A single action models an evaluation or configuration step to increase the security. A precondition represents at what point the action is selectable, and an effect models a change to the objects, predicates and functions. The first event (LHS) becomes the precondition, whilst the second event (RHS) becomes effect part of an action. This is due to the reason because LHS event occurred before than RHS in term of time. The name of an action is selected as 'LHS event-to-RHS event'. The combined list of object names from both LHS and RHS of a rule constitutes as parameters. Each domain action having one or more parameters will provide all 'entities' that are required to make any configuration change.

An additional predicate `(block_event)` is placed in each action to avoid the successive selection of the same actions, which involve the same objects. It is false before the action execution and means that the given objects have not been processed yet, and consequently rendered true to state that the same objects cannot be executed again. Notice this predicate would not restrict the repetition of actions, it would just prevent the cyclic redundancies that occur due to absence of goal state and use of maximisation metric in a problem instance (discussed later). This way the planner would not duplicate the same plan trace repeatedly to attain maximum TAA value and provide a precise set of actions.

A `total-value` function is used in the domain model to hold the accumulated TAA value of the output plan. The total-value is incremented by the TAA value of each rule upon the selection of an action. The total-value ensures that maximum number of actions (or rather complete set of actions) are provided against the identified issues based on the knowledge encoded in domain model, that would help the non-expert to

identify the security risks. In the continuing example, the temporal-association rules will be (4567 → 1234) and (1234 → 5678). For first rule, relevant object names of 4567 are *User*, *OS*, *Port* and *Process*, whereas for 1234, they are *User*, *Permission*, *Host* and *Process*. Previously the TAA value was assumed as 70%. Using this information, Figure 2 presents a complete action.

```
(:action e_4567-to-e_1234
   :parameters (?User ?OS ?Port ?Process ?Permission ?Host)
   :precondition (and (e_4567 ?User ?OS ?Port ?Process)
      (not (block_4567 ?User ?OS ?Port ?Process)))
   :effect (and (increase (total-value) 70)
      (not (e_4567 ?User ?OS ?Port ?Process))
      (block_4567 ?User ?OS ?Port ?Process)
      (e_1234 ?User ?Permission ?Host ?Process))
)
```

Figure 2: An example PDDL action, which is created using temporal-association between two events, their relevant objects/parameters and a TAA value.

Problem instances are automatically constructed, given a domain action model and event log entries of a machine that is under consideration. The instance contains objects, initial state, goal state and metric. The use of domain model ensures that the object types and initial state of problem instance are limited to the types and predicates of domain model, to prevent planning errors. The object list and their types are extracted from event log entries using object or property values (the same as described in ARM process). The goal state is kept empty as it is not known what actions need to be scheduled and there is no way of finding the missing configuration. To obtain the complete set of actions or those with the highest TAA value in the available planning time, the maximisation metric is used in the problem file. The Local search for Planning Graphs (LPG) (Gerevini et al., 2003) planner is used due to its good performance and support for PDDL (Vallati et al., 2015). The planner was executed in incremental mode to identify plans of increasing quality within a specified time. The use of AP is an integral aspect of the proposed solution due to following advantages:

1. AP algorithms provide systematic deliberation mechanisms to replace that of the human expert;
2. The parameters of domain actions provide additional information on the specified temporal link of association;
3. Objects in the plan trace belong to the underlying machine, hence providing usable results;
4. Ability to optimise based on a metric, ensuring that actions with the highest TAA values are selected to output accurate results; and
5. The knowledge is presented in standardised format and can be utilised by any PDDL supporting planner, and thus provides a benchmark domain for the AP community.

5. Empirical Analysis

This section presents an empirical analysis of the proposed solution using event logs acquired from 20 live machines, which are then evaluated against 1 machine with known security vulnerabilities. The machines 1-13 contain Microsoft Windows 7, while the machines 14-20 contains Windows 10 OS. Both set of machines have different security configurations and intended use. The purpose of this empirical analysis is to test the ability of developed technique in correctly identifying the security configuration actions that were performed on the respective machines. The reason behind selecting multiple machines to learn domain knowledge is to verify that the proposed solution is sufficiently adaptable and usable under various circumstances. The machines are configured by a administrator according to known security policies, which supplied the ground truth and made it possible to determine the accuracy of proposed solution.

5.1. Methodology

1. Extract the event logs of 20 machines and convert them into corresponding 20 object-based models.
2. The *minsup* and *minconf* values of Apriori algorithm require careful selection for eliminating 'irrelevant' rules. Choosing the appropriate values can be a difficult task, as it is very much an unsupervised process. For the empirical analysis, a wide range of values were tested for *minsup* and *minconf* ranging from $10\% - 50\%$ and $20\% - 100\%$, respectively. The results are only displayed for $minsup = 20\%$ and $minconf = 70\%$ as they showed the best results for all 20 datasets in terms of quality and quantity.
3. For each set of correlation rules, the proposed solution will produce corresponding set of temporally associated events, which will be further encoded into respective domain models.
4. A single problem instance was generated from a machine, which is known to have poor security configuration.
5. Together with the domain models and a problem instance, 20 plan traces were produced using LPG planner. To completely evaluate possible security configurations, initial state of the problem instance was manually manipulated by the expert with respect to events and different objects.
6. The correctness of an action is determined by an expert using a combined knowledge of ground truth (i.e. security policies of the 20 machines) and known security lacking of a machine from which problem instance was generated.
7. The accuracy is determined as the number of times a plan trace was able to suggest a correct sequence of actions. A sequence is considered correct if it represents the similar steps that were taken by the expert to perform a certain security task. It is calculated using the following: $correct/total\text{-}actions \times 100$.

5.2. Discussion

A live system domain action is shown in Figure 3, which was extracted from dataset-20. It shows that the configuration activity generated event 4724, which lead towards 4648 (i.e. $4724 \rightarrow 4648$). It further shows that to conduct this action, it requires the subject's Security Identifiers (SID), subject's domain names, and subject log-on ID. A live system problem file is shown in Figure 4, which was extracted from a test machine having poor security configurations. The extraction process is automatic, and only requires (1) event log of the test machine and (2) the domain model containing expert knowledge from other machine. The objects of problem file are the object and property values of event log entries, whereas the object types are the object names defined in the corresponding event log entries. The initial state consists of those events, which are common between the given domain model and event log file. The input parameters are taken from the domain model, which are substituted by respective values (objects) of a problem file. Based on the domain and problem file, a plan trace is generated by using the LPG planner and shown in Figure 5. This provides a sequences of actions that can be used by a non-expert to improve the security of 'unconfigured' machines. By providing event object values as parameters in problem file, the plan trace outputs in-process and to-the-point information about the underlying machine, which can be quite useful in resolving the identified vulnerabilities. It provides an additional layer of information that will also be used to develop an automated vulnerability mitigation solution in future work.

As the plan trace is difficult to understand by non-experts, we also created a simple plan parser (SPP) tool that uses event log file and domain model to elaborate actions of plan trace regarding event type IDs and objects. An example output of the first action of plan trace is shown in Figure 6. A complete explanation of the plan trace is presented in Figure 7. It should be noticed here that the sequences of events generated from a single dataset are not universally applicable; they all depend on the set of activities performed or conscious decisions taken by an administrator. Even though the event log entries are generated in linear manner, there is no way of knowing which of them are connected with respect to a single human expert action. The solution presented in this paper can be used to find those sequences and connections in the presence of large amount of noise, and facilitate the autonomous replication of human expert actions into a domain model. Following is the brief interpretation of planner output in terms of events relationships shown in Figure 7:

```
(:action e_4724-to-e_4648
   :parameters (?SubjectUserSid - SubjectUserSid ?TargetDomainName - TargetDomainName
      ?TargetSid - TargetSid ?TargetUserName - TargetUserName ?SubjectUserName - SubjectUserName
      ?SubjectDomainName - SubjectDomainName ?SubjectLogonId - SubjectLogonId)
   :precondition (and (e_4724 ?SubjectUserSid ?TargetDomainName ?TargetSid ?TargetUserName)
      (not (block_4724 ?SubjectUserSid ?TargetDomainName ?TargetSid ?TargetUserName)))
   :effect (and (increase (total-value) 99)
      (not (e_4724 ?SubjectUserSid ?TargetDomainName ?TargetSid ?TargetUserName))
      (block_4724 ?SubjectUserSid ?TargetDomainName ?TargetSid ?TargetUserName)
      (e_4648 ?SubjectDomainName ?SubjectLogonId ?SubjectUserName ?SubjectUserSid))
)
```

Figure 3: An action from domain model with a TAA value of 99, which was extracted from dataset-20 temporal-association rules.

```
(define (problem problem_Events)
   (:domain EventRelations)
   (:objects
      USER1MSI - SubjectDomainName          0_0x3e7 - SubjectLogonId
      USER1MSI$ - SubjectUserName           S_1_5_18 - SubjectUserSid
      0_0 - PuaCount                        0_0xc44e - PuaPolicyId
      User1SMSI - TargetDomainName          S_1_5_32_568 - TargetSid
      USER2MSI - TargetUserName             USER1MSI - Workstation
   )
   (:init (= (total-value) 0)   (e_4717 USER1MSI 0_0x3e7 USER1MSI$ S_1_5_18) )
   (:goal (and ) )
   (:metric maximize (total-value))
)
```

Figure 4: A simplified version of problem file generated automatically from a machine that has poor security configurations.

```
0:  (E_4717-TO-E_4797 USER1MSI 0_0X3E7 USER1MSI S_1_5_18 USER2MSI USER1MSI)
1:  (E_4797-TO-E_4902 0_0X3E7 S_1_5_18 USER2MSI USER1MSI 0_0 0_0XC44E)
2:  (E_4902-TO-E_4904 0_0 0_0XC44E USER1MSI 0_0X3E7 USER1MSI S_1_5_18)
3:  (E_4904-TO-E_4905 S_1_5_18 USER1MSI 0_0X3E7 USER1MSI S_1_5_18)
4:  (E_4905-TO-E_4724 USER1MSI 0_0X3E7 USER1MSI S_1_5_18 S_1_5_18 USER1SMSI S_1_5_32_568 USER2MSI)
5:  (E_4724-TO-E_4648 S_1_5_18 USER1MSI S_1_5_32_568 USER2MSI USER1MSI 0_0X3e7 USER1MSI)
```

Figure 5: Action plan extracted for the test machine using the domain knowledge acquired from dataset-20.

- *Identification of Vulnerability* – Event 4717 shows that a certain user, with SID as *S_1_5_18* and username as *USER1MSI*, successfully logged on a work-station. The login occurred with special privileges. After that, event 4797 shows that the user queried the existence of a blank password for an account and found one with username as *USER2MSI*.
- *Benign event* – On every login, some routine events are subsequently triggered depending on the audit policy. In this case, such events show that the policy table was created (event 4902) and custom security events sources were registered/unregistered (events 4904 and 4905).

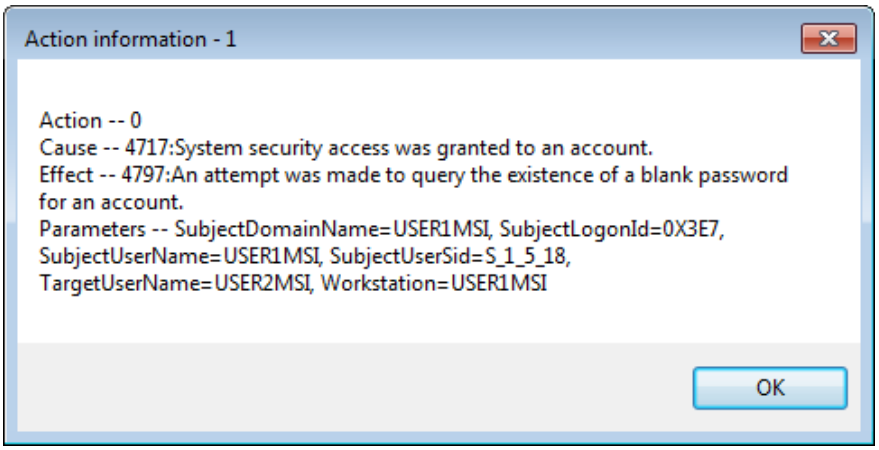

Figure 6: Results from parsing the first action of plan trace using SPP. It provides and improved understanding of each plan trace action and also facilitated in determining accuracy.

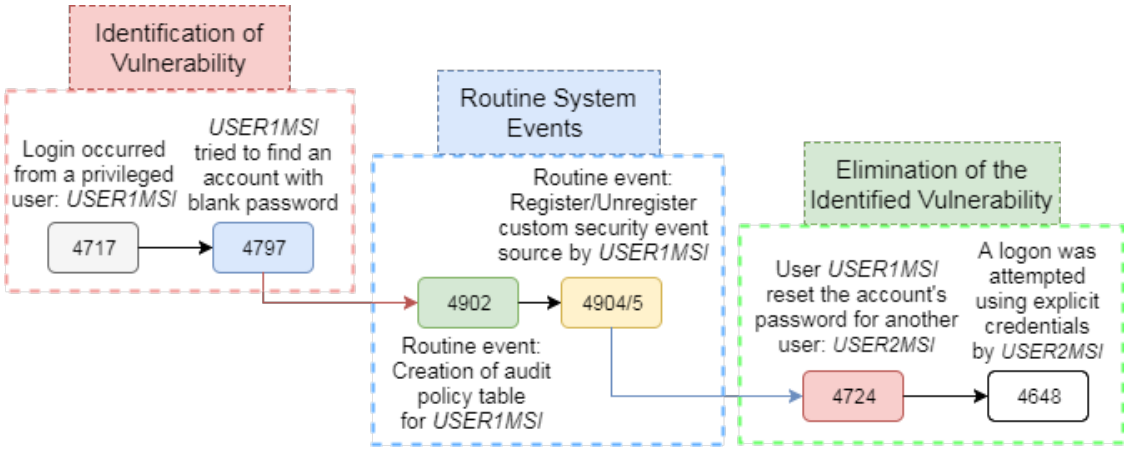

Figure 7: Elaboration of the sequence of events, extracted in the form of plan trace using dataset-20 domain model. The related events and corresponding objects reflect the security configuration actions of human expert on a particular machine.

- *Elimination of Vulnerability* – After identifying blank password of a user *USER2MSI*, the event 4724 depicts the change of account password. The objects of plan trace show that the user (subject) with SID *S_1_5_18* reset the password for another user (target) with SID as *S_1_5_32_568*. After that, the user *USER2MSI* logged in as described by event 4648.
- The sequence of events shows that an administrator searched for the user accounts having no password, meanwhile performing some routine activities. Upon finding such an account, the user password was changed to strengthen the account, hence securing the overall system.

Another in-process plan trace is presented in Figure 8, which provides the incorrect set of actions extracted using dataset-17. The plan trace parameters are as following: *USERWORK* is the subject user name, *DESKTOP_61D3AOQ* is the machine name, *S_1_5_19* is the subject user SID and *O_443* is the port number (i.e., Universal TCP Firewall Port). At first, the events 4663 and 4670 show that a certain object was accessed and its permissions were modified. Following that, the event 5152 was triggered which means that the system's firewall blocked a network packet. At the end, the event 5145 shows that the a certain network share object was checked to see whether client can be granted desired access. This plan trace is incomplete and has erroneous ordering of events. It has linked two individual administrative tasks (access

permission control of an object and network filtering by a firewall). These actions do not provide any useful knowledge to the non-experts that might improve their system's security.

```
0:  (E_4663-TO-E_4670 USERWORK DESKTOP_61D3AOQ S_1_5_19)
1:  (E_4670-TO-E_5152 USERWORK DESKTOP_61D3AOQ S_1_5_19 O_443)
2:  (E_5152-TO-E_5145 O_443 USERWORK DESKTOP_61D3AOQ S_1_5_19)
```

Figure 8: Incorrect set of actions extracted for the test machine using the domain knowledge acquired from dataset-17.

The empirical analysis presented in this paper was performed on an Intel i7 3.50GHz processor running Windows 7 OS with 16GB RAM. The results are presented from each layer of processing in Table 1, such as number of event logs, correlation rules, temporal-association rules, domain actions etc. It is noticeable that the plan traces have shown suitable accuracy (ranging from $62\% - 91\%$) and performance (around 8 minutes for processing 31,729 log entries). There is potential for a degree of uncertainty in finding the accuracy as the expert does not know the complete set of events generated when certain security configuration activities are performed on a machine. Moreover there can be one or more ways of executing the same task, which will produce a different set of event entries. For the same reasons, *true and false negatives* were not included in determining TAA value due to ambiguity in what constitutes a negative. There is also uncertainty around the plan traces as LPG is a sub-optimal planner, but its ability to apply stochastic heuristics to draw plan traces without any goal state motivated the use. Furthermore, any successfully found plan trace will provision the non-expert with one of the many ways to conduct a certain configuration.

The higher amount of unique entries implies substantial user activity and should result in more domain actions. Table 1 shows that the maximum number of event log entries (31,729 with 20 unique event) were identified in dataset-20 and resulted in 13 domain actions in 6.77 minutes. On the contrary, the minimum number of entries (3,404 with 33 unique events) were found in dataset-19 and produced 46 domain actions. This dataset has the maximum quantity among other datasets and required 43 seconds to process. This shows that higher number of unique entries produced larger object-based model, and therefore resulted in more number of domain actions. Hence there is a proportional relationship between the number of unique events and domain actions, which demonstrates the 'completeness' of the proposed solution.

The number of correlation rules is not always proportional to the number of domain actions and domain extraction time. The maximum number of association rules (281,508,612) is generated from dataset-3, but only resulted in 10 rules and 27 actions in 4.5 minutes. On the other hand, dataset-19 only used 114,728 association rules to output 46 domain actions in 6.77 minutes. Due to temporal validation on each step, the redundant and incorrect event relationships are filtered, and the solution produces useful results in reasonable time. Furthermore the elimination of unwanted rules on each phase reduces the overall processing time. This signifies the 'efficiency' of the proposed solution.

It has also been observed that datasets with lower accuracy contain such rules, where a single event is linked to many others. Here Automated Planning plays an important role to determine the suitable set of actions with respect to TAA values. It should also be noticed that the solution does not create domain action where they the temporal-association rules are not present, regardless of number of entries in event logs. Four such datasets were excluded from the testing as they did not produce any domain actions. The object-based model ensures that a rule, and consequently a domain action, is created only where there is a relation among the properties of event log entries. Apart from the results, it is important to consider that the solution will provide improved accuracy if the configuration, environment and intended use of target computer are same as of the machine from which the knowledge was extracted. The similar settings and context will make it easier to match any learnt knowledge and help the user perform any necessary changes.

Event log dataset	No. of events	No. of unique events	No. of correlation rules	No. of event-based rules	No. of subsets	No. of temporal-association connections	Maximum TAA value	Domain extraction duration	No. of actions in domain	Planner execution time	No. of actions in plan	Accuracy %
Event Acquisition			*ARM, Temporal-Association Rules and Domain Modelling*						*Automated Planning*			
1	7,756	33	13,423,448	23	47	7	96.07	01m:47s	15	15.33s	56	89
2	8,254	33	6,629,475	20	43	6	84.52	01m:50s	12	33.12s	57	78
3	11,956	29	281,508,612	24	64	10	79.67	04m:29s	27	27.98s	46	74
4	7,729	32	11,559,989	22	47	6	86.56	01m:44s	12	33.83s	47	78
5	7,710	32	17,355,471	28	53	7	100	01m:47s	14	18.33s	67	84
6	7,888	32	13,424,959	23	47	7	95.72	01m:47s	15	27.27s	47	88
7	8,262	32	12,200,902	23	47	7	93.77	01m:51s	16	54.45s	77	86
8	9,721	32	29,597,902	23	70	11	86.21	02m:18s	26	10.78s	71	77
9	10,206	31	27,839,778	24	69	11	67.4	02m:23s	26	52.50s	53	62
10	7,785	32	17,545,864	23	47	7	92.36	01m:48s	15	58.11s	55	81
11	9,862	33	28,954,117	24	68	11	67.33	02m:20s	22	41.19s	45	63
12	9,771	31	32,736,160	24	64	9	90.94	02m:20s	25	59.92s	48	87
13	7,577	31	16,438,253	23	46	6	100	01m:46s	10	0.39s	14	95
14	13,027	32	1,132,135	41	101	8	100	02m:48s	27	57.66s	72	79
15	10,948	15	12,472,706	4	16	5	100	02m:24s	5	0.03s	6	91
16	8,836	24	9,049,482	19	78	8	67.75	01m:56s	31	114.33s	52	63
17	26,008	17	714,287	11	17	5	100	05m:36s	8	0.001s	6	80
18	3,594	23	1,532,780	17	13	6	100	00m:46s	6	52.81s	5	63
19	3,404	33	114,728	15	259	18	91.33	00m:43s	46	59.00s	92	85
20	31,729	20	778,354	13	30	6	66.59	06m:44s	13	54.61s	32	64

Table 1: Results from conducting empirical analysis during domain acquisition and automated planning. The analysis is performed on 20 event log datasets from live machines and shows the output numbers from each stage of processing.

6. Conclusion

This paper presents a novel automated technique to extract domain action models from security event logs without any human intervention. The main purpose of this solution is to enable non-experts to conduct expert analysis of the new or unseen machine without spending significant amount of time and effort in acquiring security knowledge. The technique is based on a scenario where an expert performs the security evaluation or configuration on a system. Every change or action will be recorded in the form of event log entries. Identifying temporal-association relationships among such event log entries in an automated manner can provide a sequence of actions that expert took to reform the system security. In addition, storing and representing the extracted knowledge in a standardised PDDL format will increase the applicability of the

solution due to the wide presence and understanding of automated planners.

The presented technique was developed and tested using 20 event log datasets. First, it creates an object-based model to represent event log entries and extracts strong correlation rules using ARM techniques. The correlation rules are then formulated and validated into sequences of temporally correlated rules by using frequency and temporal metrics. Knowledge extracted in the form of temporal-association rules is encoded into a domain action model using PDDL. The developed solution automatically generates problem instances as well by encoding live events and objects from underlying machine. Given the domain and problem files, LPG planner is used to extract plan traces. Despite the event log entries are produced in high frequency and might contain large amount noise (benign events), the proposed solution demonstrated that it can successfully extract the domain knowledge and be applicable in practical environments. Future work includes the augmentation of SPP to translate the output sequence of events into a step-by-step guide involving tools and techniques and reduce the manual effort of implementing security measures.

References

Agrawal, R., Imieliński, T., Swami, A., 1993. Mining association rules between sets of items in large databases. In: Acm sigmod record. Vol. 22. ACM, pp. 207–216.

Agrawal, R., Srikant, R., et al., 1994. Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB. Vol. 1215. pp. 487–499.

Ale, J. M., Rossi, G. H., 2000. An approach to discovering temporal association rules. In: Proceedings of the 2000 ACM Symposium on Applied computing-Volume 1. ACM, pp. 294–300.

Amos-Binks, A., Clark, J., Weston, K., Winters, M., Harfoush, K., 2017. Efficient attack plan recognition using automated planning. In: Computers and Communications (ISCC), 2017 IEEE Symposium on. IEEE, pp. 1001–1006.

Angelov, P. P., 2013. Autonomous learning systems: from data streams to knowledge in real-time.

Backes, M., Hoffmann, J., Künnemann, R., Speicher, P., Steinmetz, M., 2017. Simulated penetration testing and mitigation analysis. arXiv preprint arXiv:1705.05088.

Bechlivanidis, C., Lagnado, D. A., 2016. Time reordered: Causal perception guides the interpretation of temporal order. Cognition 146, 58–66.

Bettinsoli, M. L., Maass, A., Kashima, Y., Suitner, C., 2015. Word-order and causal inference: The temporal attribution bias. Journal of Experimental Social Psychology 60, 144–149.

Bleisch, S., Duckham, M., Galton, A., Laube, P., Lyon, J., 2014. Mining candidate causal relationships in movement patterns. International Journal of Geographical Information Science 28 (2), 363–382.

Botea, A., Enzenberger, M., Müller, M., Schaeffer, J., 2005. Macro-ff: Improving ai planning with automatically learned macro-operators. Journal of Artificial Intelligence Research 24, 581–621.

Box, G. E., Jenkins, G. M., Reinsel, G. C., Ljung, G. M., 2015. Time series analysis: forecasting and control. John Wiley & Sons.

Cresswell, S. N., McCluskey, T. L., West, M. M., 2013. Acquiring planning domain models using locm. The Knowledge Engineering Review 28 (2), 195–213.

Gal, A., 2003. Evaluating matching algorithms: the monotonicity principle. In: Semantic Integration Workshop (SI-2003). p. 167.

Gerevini, A., Saetti, A., Serina, I., 2003. Planning through stochastic local search and temporal action graphs in lpg. Journal of Artificial Intelligence Research 20, 239–290.

Ghallab, M., Nau, D., Traverso, P., 2004. Automated Planning: theory and practice. Elsevier.

Gregory, P., Cresswell, S., 2015. Domain model acquisition in the presence of static relations in the lop system. In: ICAPS. pp. 97–105.

Hipp, J., Güntzer, U., Nakhaeizadeh, G., 2000. Algorithms for association rule mininga general survey and comparison. ACM sigkdd explorations newsletter 2 (1), 58–64.

Hoffmann, J., 2015. Simulated penetration testing: From" dijkstra" to" turing test++". In: ICAPS. pp. 364–372.

Ishibuchi, H., Kuwajima, I., Nojima, Y., 2007. Prescreening of candidate rules using association rule mining and pareto-optimality in genetic rule selection. In: Knowledge-Based Intelligent Information and Engineering Systems. Springer, pp. 509–516.

Jilani, R., Crampton, A., Kitchin, D., Vallati, M., 2015. Ascol: A tool for improving automatic planning domain model acquisition. In: Congress of the Italian Association for Artificial Intelligence. Springer, pp. 438–451.

Karthikeyan, T., Ravikumar, N., 2014. A survey on association rule mining. International Journal of Advanced Research in Computer and Communication Engineering 3 (1), 2278–1021.

Khan, S., Parkinson, S., 2017. Towards automated vulnerability assessment.

Khan, S., Parkinson, S., Qin, Y., Aug 2017. Fog computing security: a review of current applications and security solutions. Journal of Cloud Computing 6 (1), 19.
URL https://doi.org/10.1186/s13677-017-0090-3

Liang, Z., Xinming, T., Lin, L., Wenliang, J., 2005. Temporal association rule mining based on t-apriori algorithm and its typical application. In: Proceedings of International Symposium on Spatio-temporal Modeling, Spatial Reasoning, Analysis, Data Mining and Data Fusion.

Long, K., Radhakrishnan, J., Shah, R., Ram, A., 2009. Learning from human demonstrations for real-time case-based planning.
Mai, T., Vo, B., Nguyen, L. T., 2017. A lattice-based approach for mining high utility association rules. Information Sciences 399, 81–97.
McCluskey, T. L., Cresswell, S., Richardson, N. E., West, M. M., 2009. Action knowledge acquisition with opmaker2. In: International Conference on Agents and Artificial Intelligence. Springer, pp. 137–150.
McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D., 1998. Pddl-the planning domain definition language.
Mikhaylov, D. V., Kozlov, A. P., Emelyanov, G. M., 2017. An approach based on analysis of n-grams on links of words to extract the knowledge and relevant linguistic means on subject-oriented text sets. Computer Optics 41 (3), 461–471.
Miklossy, J., 2011. Alzheimer's disease-a neurospirochetosis. analysis of the evidence following koch's and hill's criteria. Journal of neuroinflammation 8 (1), 90.
Morgan, S. L., Winship, C., 2014. Counterfactuals and causal inference. Cambridge University Press.
Mushtaq, M. F., Akram, U., Khan, I., Khan, S. N., Shahzad, A., Ullah, A., 2017. Cloud computing environment and security challenges: A review. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS 8 (10), 183–195.
Natarajan, S., Bangera, V., Khot, T., Picado, J., Wazalwar, A., Costa, V. S., Page, D., Caldwell, M., 2017. Markov logic networks for adverse drug event extraction from text. Knowledge and Information Systems 51 (2), 435–457.
Nazerfard, E., Rashidi, P., Cook, D. J., 2011. Using association rule mining to discover temporal relations of daily activities. In: International Conference On Smart homes and health Telematics. Springer, pp. 49–56.
Park, J. S., Chen, M.-S., Yu, P. S., 1995. An effective hash-based algorithm for mining association rules. Vol. 24. ACM.
Parkinson, S., Somaraki, V., Ward, R., 2016. Auditing file system permissions using association rule mining. Expert Systems with Applications 55, 274 – 283.
URL http://www.sciencedirect.com/science/article/pii/S0957417416300586
Pearl, J., 2009. Causality: models, reasoning, and inference, 2nd Edition. Cambridge University Press, Cambridge.
Piatetsky-Shapiro, G., 1991. Discovery, analysis and presentation of strong rules. Knowledge discovery in databases, 229–238.
Riabov, A., Sohrabi, S., Udrea, O., Hassanzadeh, O., 2016. Efficient high quality plan exploration for network security. In: 11th Scheduling and Planning Applications woRKshop(SPARK).
Roddick, J. F., Spiliopoulou, M., 2002. A survey of temporal knowledge discovery paradigms and methods. IEEE Transactions on Knowledge and data engineering 14 (4), 750–767.
Sahoo, R. K., Oliner, A. J., Rish, I., Gupta, M., Moreira, J. E., Ma, S., Vilalta, R., Sivasubramaniam, A., 2003. Critical event prediction for proactive management in large-scale computer clusters. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp. 426–435.
Sarma, P. K. D., Mahanta, A. K., 2012. Reduction of number of association rules with inter itemset distance in transaction databases. International Journal of Database Management Systems 4 (5), 61.
Schauland, D., Jacobs, D., 2016. Managing the windows event log. In: Troubleshooting Windows Server with PowerShell. Springer, pp. 17–33.
Shah, M., Chrpa, L., Jimoh, F., Kitchin, D., McCluskey, T., Parkinson, S., Vallati, M., 2013. Knowledge engineering tools in planning: State-of-the-art and future challenges. Knowledge Engineering for Planning and Scheduling 53.
Shahaf, D., Amir, E., 2006. Learning partially observable action schemas. In: Proceedings of the National Conference on Artificial Intelligence. Vol. 21. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 913.
Simache, C., Kaâniche, M., Saidane, A., 2002. Event log based dependability analysis of windows nt and 2k systems. In: Dependable Computing, 2002. Proceedings. 2002 Pacific Rim International Symposium on. IEEE, pp. 311–315.
Singer, J. D., Willett, J. B., 2003. Applied longitudinal data analysis: Modeling change and event occurrence. Oxford university press.
Sliva, A., Reilly, S. N., Casstevens, R., Chamberlain, J., 2015. Tools for validating causal and predictive claims in social science models. Procedia Manufacturing 3, 3925–3932.
Tan, T.-F., Wang, Q.-G., Phang, T.-H., Li, X., Huang, J., Zhang, D., 2015. Temporal association rule mining. In: International Conference on Intelligent Science and Big Data Engineering. Springer, pp. 247–257.
Treur, J., 2016. Dynamic modeling based on a temporal–causal network modeling approach. Biologically Inspired Cognitive Architectures 16, 131–168.
Vaarandi, R., 2002. Sec-a lightweight event correlation tool. In: IP Operations and Management, 2002 IEEE Workshop on. IEEE, pp. 111–115.
Valenzano, R. A., Sturtevant, N., Schaeffer, J., Buro, K., Kishimoto, A., 2010. Simultaneously searching with multiple settings: An alternative to parameter tuning for suboptimal single-agent search algorithms. In: Third Annual Symposium on Combinatorial Search.
Vallati, M., Chrpa, L., Grzes, M., McCluskey, T. L., Roberts, M., Sanner, S., 2015. The 2014 international planning competition: Progress and trends. AI Magazine 36 (3), 90–98.
Vaquero, T., Tonaco, R., Costa, G., Tonidandel, F., Silva, J. R., Beck, J. C., 2012. itsimple4. 0: Enhancing the modeling experience of planning problems. In: System Demonstration–Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12). pp. 11–14.
Verma, K., Vyas, O. P., Vyas, R., 2005. Temporal approach to association rule mining using t-tree and p-tree. In: International Workshop on Machine Learning and Data Mining in Pattern Recognition. Springer, pp. 651–659.
Wang, L., Meng, J., Xu, P., Peng, K., 2018. Mining temporal association rules with frequent itemsets tree. Applied Soft Computing 62, 817–829.
Winarko, E., Roddick, J. F., 2007. Armada–an algorithm for discovering richer relative temporal association rules from interval-

based data. Data & Knowledge Engineering 63 (1), 76–90.
Wu, K., Yang, Q., Jiang, Y., 2005. Arms: Action-relation modelling system for learning action models. Proceedings of ICKEPS.
Zhuo, H. H., 2015. Crowdsourced action-model acquisition for planning. In: AAAI. pp. 3439–3446.
Zhuo, H. H., Kambhampati, S., 2013. Action-model acquisition from noisy plan traces. In: IJCAI. pp. 2444–2450.